Data Mining Lab3(K-means Clustering)

//作者: Bo-Wen Duan (Paul)

//聯絡方式: [bowenduan618@gmail.com](mailto:bowenduan618@gmail.com)

//修改日期: 2016 1/10

* 實際資料庫: iris.arff
* 輸出檔案: KmeansResult.txt
* 執行檔: KmeansClustering.java
* 將資料夾”DataMininghw3”放置eclipse專案區中並匯入執行(或使用cmd編譯及執行KmeansClustering.java檔)。
* K-means 演算法的整個流程：

1. 首先從分群對像中隨機選出K個對像作為群集的質心(初始參數的K代表分群結果的群集數)。
2. 對剩餘的每個對象，根據它們分別到這個K個質心的距離，將它們指定到最相似的群集(因為K-means是利用距離來量化相似度的，所以我們這裡可以理解為是”將它們指定到離最近最近距離的質心所屬的群集”）。
3. 然後重新計算質心位置。
4. 以上過程不斷反複，直到準則函數收斂為止。
5. 通常採用平方誤差準則，定義如下：
   * 其中，SSE代表的意思是所有群集中各對像到​​其所屬群集的質心平方誤差和.
   * K: 分群結果群集個數
   * Ci: 第i個群集
   * x ：群集中分群對象
   * mi: 第i個群集的質心

* K-means的優點和不足：

1. 能處理大型資料集，結果群集相當緊湊，並且群集和群集之間明顯分離。
2. 計算複雜性O(tkn) t:迭代次數、K：分群數 n:樣本數。
3. 該演算法必須事先給定群集數和質新，群集數和質心的初始值設定往往會對分群的演算法影響較大。
4. 通常會在獲得一個局部最優值時停止。
5. 並且只適合對數值型資料分群。
6. 只適用於分群結果為球狀的資料庫，K-means方法不適合發現非球狀的分群，或者大小差別很大的群集。
7. 對噪音(noise)和孤立點數據敏感，少量的該類數據對質心的計算會產生極大的影響。

* K-Means程式碼(KmeansClustering.java)

程式主要結構:

* + Import檔: java.io.\*

java.util.\*

* + 主類別: KmeansClustering
    - 靜態變數: InputFileName //輸入檔名

OutputFileName //輸出檔名

* + - 函式: main() //程式進入點

sop() //列印方法

* 子類別: KMeans
  + 變數: TestData //儲存原始資料。

DisTestData //儲存每一筆資料與原點之間的距離平方。

NumOfClusters //儲存分群數。

TestCases //儲存資料筆數。

TestAttributes //儲存每筆資料的屬性數。

Centroid //儲存分群後每群的質心座標。

AryResultClusters //儲存分群後每一群的資料筆編號。

ClusterNo //儲存分群後每一群的個數。

* 函式: GetNumOfClusters() //分群數方法

SetNumOfClusters(int mNumOfClusters) //設定分群數方法。

GetTestData() //取得原始資料方法。

GetTestCases() //取得資料筆總數方法。

GetTestAttributes() //取得資料筆屬性數方法。

GetClusterNo() //取得分群後每一群集的資料筆數方法。

GetCentroid() //取得初始質心座標方法。

GetDisTestData() //取得所有資料筆與原點距離平方方法。

KMeans(double[][] input\_data) //建構子

GetStartingPointArrayIndex() //取的初始質心index方法。

EuclideanDistance(double[][] centroid) //計算歐幾里得距離方法。

DoPartition() //分群方法。

GetNumOfObservations() //取得分群後每一群集的資料筆數方法。

GetAryResultClusters() //取得分群後每一群集的資料筆編號方法。

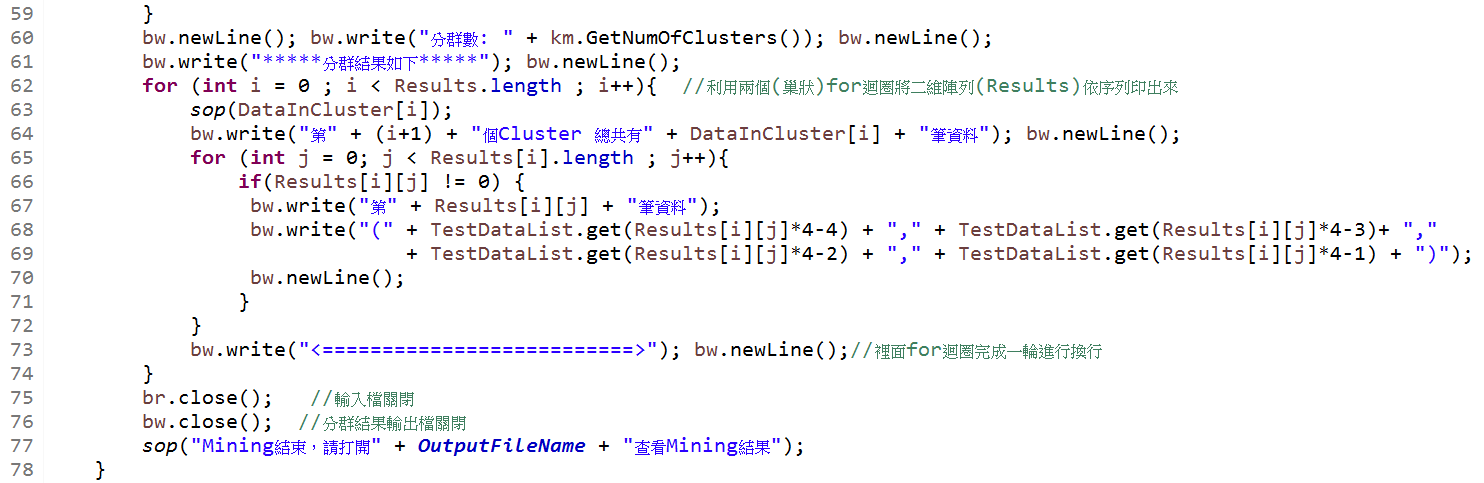
* Import檔
* java.io.\*: 檔案IO所需的套件包。
* java.util.\*: 集合所需的套件包。

(可視需求import會用到的子類別進來就好)

* 主類別KmeansClustering-靜態變數

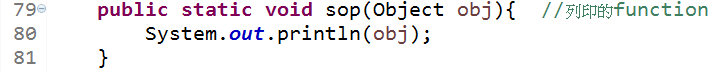


* 第25行: private final static String **InputFileName** = "retail.txt"; 設置要輸入的檔案名稱。
* 第26行: private final static String **OutputFileName** = "MiningResult.txt"; 設置Mining的結果要寫出的檔案名稱。
* 主類別KmeansClustering-函式
* main()



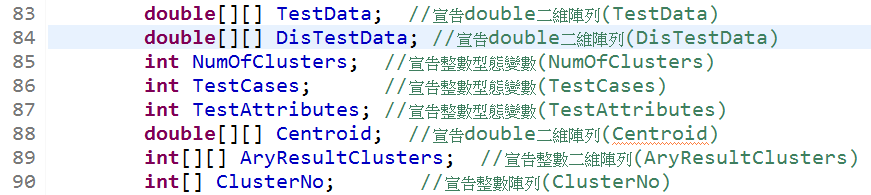
1. 第27行: 程式的進入點，throws IOException為輸入輸出丟出例外。
2. 第28~29行: 宣告讀寫檔案以及緩衝讀寫的變數，處理檔案輸入以及輸出文字型態(Big5)。
3. 第30行: 宣告字串形態變數(Line,TempString)。
4. 第31行: 宣告字串陣列TempArray，陣列大小為資料筆的屬性數。
5. 第32行: 宣告一個ArrayList(TestDataList)泛型為字串。
6. 第33~39行: 使用while迴圈將資料一行一行讀進來，用Line接收每一行資料，直到讀到的是null則跳出迴圈。用TempString接收Line的每一行內容。將TempString(每一行資料)以","作為切割點進行切割，存到TempArray陣列中。最後用for迴圈將TempArray的資料依序存到TestDataList中。
7. 第40行: 宣告整數型態變數TestDataCount，為TestDataList除以資料筆屬性個數(資料筆個數)。
8. 第41行: 宣告整數型態變數count為0。
9. 第42行: 宣告double型態二維陣列(TestData)，陣列大小為資料筆個數，長度為每個資料筆的屬性個數。
10. 第43~48行: 利用兩個(巢狀)for迴圈，利用count作為index取出TestDataList的字串，再將字串轉換成double依序存到二維陣列中。
11. 第49行: 宣告一個KMeans類別變數km，將TestData作為引數傳入。
12. 第50行: 利用KMeans類別變數km，呼叫SetNumOfClusters方法設定分群數。
13. 第51行: 宣告整數型態二維陣列(Results)，接收km的DoPartition()所回傳的整數型態二維陣列(每一群集的資料筆編號)。
14. 第52行: 宣告double型態二維陣列(TestDataDistance)，接收km的GetDisTestData()所回傳的double型態二維陣列(每一資料筆分別與原點之間的距離平方)。
15. 第53行: 宣告整數型態陣列(DataInCluster)，接收km的GetNumOfObservations()所回傳的整數型態陣列(每一群集的資料筆數)。
16. 第54~74行: 將Mining結果(原始資料輸入檔名，每一資料筆分別與原點之間的距離平方，分群數，每一群集的資料筆數，每一群集的資料筆編號以及座標)寫出到設定的檔案。
17. 第75~76行: 寫入以及寫出的檔案關閉。
18. 第77行: 列印Mining結束的訊息。

* sop()



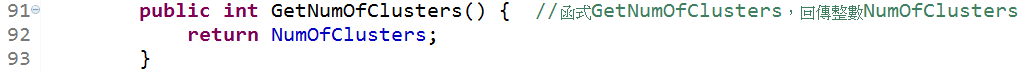
1. 第79~81行: sop()函式執行列印功能。

* 子類別KMeans -變數



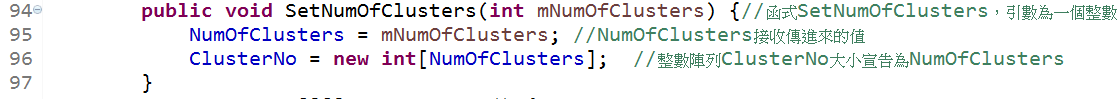
1. 第83行: 宣告double二維陣列(TestData)，儲存原始資料。
2. 第84行: 宣告double二維陣列(DisTestData)，儲存每一筆資料與原點之間的距離平方。
3. 第85行: 宣告整數型態變數(NumOfClusters)，儲存分群數。
4. 第86行: 宣告整數型態變數(TestCases)，儲存資料筆數。
5. 第87行: 宣告整數型態變數(TestAttributes)，儲存每筆資料的屬性數。
6. 第88行: 宣告double二維陣列(Centroid)，儲存分群後每群的質心座標。
7. 第89行: 宣告整數二維陣列(AryResultClusters)，儲存分群後每一群的資料筆編號。
8. 第90行: 宣告整數陣列(ClusterNo)，儲存分群後每一群的個數。

* 子類別KMeans -函式
* GetNumOfClusters()



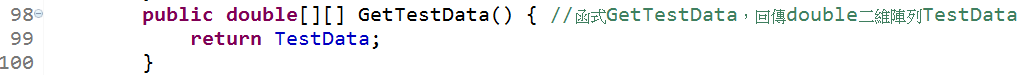
1. 第91~93行: GetNumOfClusters()為一個函式，回傳整數NumOfClusters(分群數)。

* SetNumOfClusters(int mNumOfClusters)



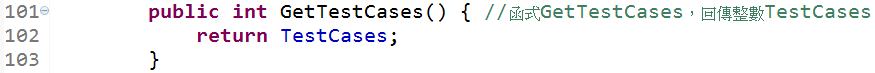
1. 第94~97行: SetNumOfClusters(int mNumOfClusters)為一個函式，引數為一個整數。NumOfClusters接收傳進來的值。整數陣列ClusterNo大小宣告為NumOfClusters。(設定分群個數方法)

* GetTestData()



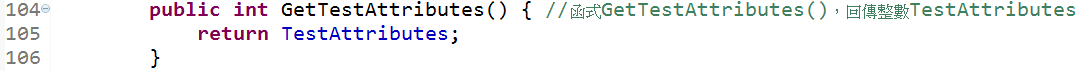
1. 第98~100行: GetTestData()為一個函式，回傳double二維陣列TestData。(取得原始資料陣列方法)

* GetTestCases()



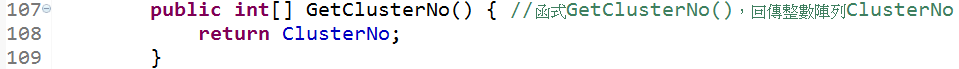
1. 第101~103行: GetTestCases()為一個函式，回傳整數TestCases。(取得原始資料筆數方法)。

* GetTestAttributes()



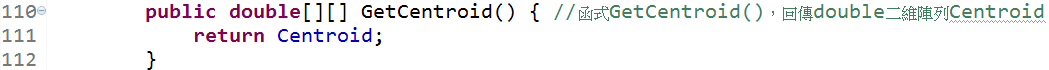
1. 第104~106行: GetTestAttributes()為一個函式，回傳整數TestAttributes。(取得每筆資料屬性數方法)。

* GetClusterNo()



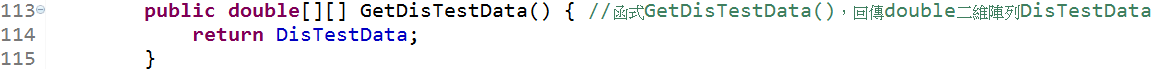
1. 第107~109行: GetClusterNo()為一個函式，回傳整數陣列ClusterNo。(取得分群後每群的資料筆數方法)。

* GetCentroid()



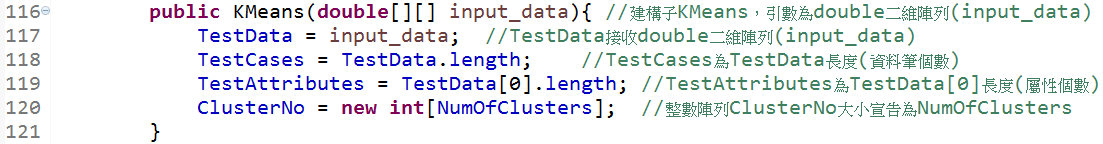
1. 第110~112行: GetCentroid()為一個函式，回傳double二維陣列Centroid。(取得分群後每群的質心座標方法)。

* GetDisTestData()



1. 第113~115行: GetDisTestData()為一個函式，回傳double二維陣列DisTestData。(取得每筆資料與原點之間距離平方的方法)。

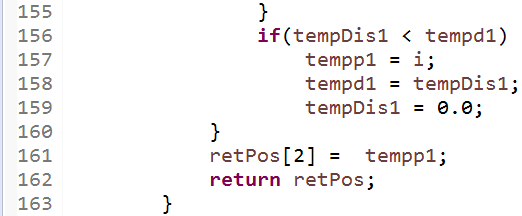
* KMeans(double[][] input\_data)



1. 第116~121行: 建構子KMeans，引數為double二維陣列(input\_data)。TestData接收double二維陣列(input\_data)。TestCases為TestData長度(資料筆個數)。TestAttributes為TestData[0]長度(屬性個數)。整數陣列ClusterNo大小宣告為NumOfClusters。

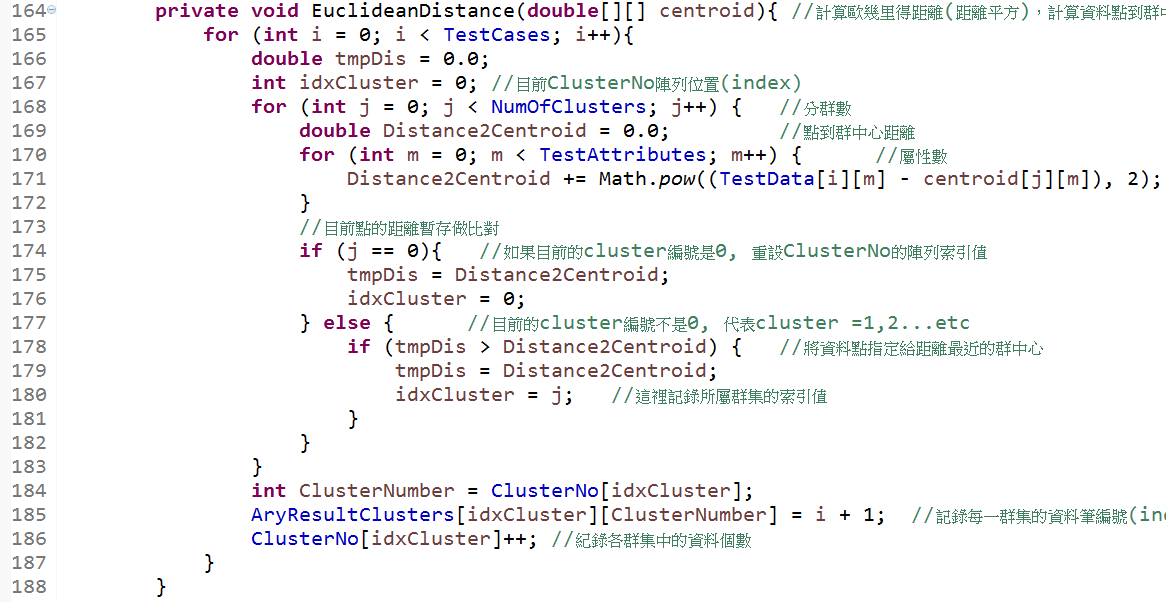
* GetStartingPointArrayIndex()





1. 第122行: GetStartingPointArrayIndex()為一個函式，回傳型態為一個整數陣列(儲存分群初始質心點的index)。
2. 第123行: 宣告一個double二維陣列(testDataDistance)，大小為資料筆數，長度為2
3. 第124行: 宣告double型態變數(SumData)，儲存資料筆與原點之間的距離平方，初始值設為0.0。
4. 第125~132行: 利用巢狀for迴圈，外層是資料筆數，內層是屬性數，將每一筆資料的每個屬性減0後平方，再加到SumData，最後得到每一筆資料與原點之間的距離平方。testDataDistance[m][0]儲存資料筆的索引，testDataDistance[m][1]儲存資料與原點之間的距離平方。
5. 第133行: DisTestData接收計算完的testDataDistance。
6. 第134行: 宣告一個整數陣列(rePos)，大小為分群數。
7. 第135行: 宣告一個整數型態變數(p1)，為隨機取一個資料筆數索引(double強轉int)。
8. 第136行: 宣告兩個整數型態變數(tempp,tempp1)，分別給定值0。
9. 第137行: 宣告兩個double型態變數(tempd,tempd1)，tempd給定值0.0，tempd1給定值為double型態的最大值。
10. 第138行: rePos[0]放入值p1。(第1個點隨機取)
11. 第139~149行: 利用巢狀for迴圈，外層是資料筆數，內層是屬性數，tempDis接收每一筆資料分別與rePos[0]索引所對應的資料計算距離。再利用if迴圈判斷若tempDis大於tempd則tempd改為tempDis，tempp改為當下外層for迴圈的指標，tempDis再歸零再進行下一輪，整個跑完最後的tempp放入rePos[1]。(第2個點要離第1個點最遠)
12. 第150~162行: 利用巢狀for迴圈，外層是資料筆數，內層是屬性數，tempDis1接收每一筆資料分別與rePos[0]和rePos[1]索引所對應的資料計算距離相加。再利用if迴圈判斷若tempDis1小於tempd1則tempd1改為tempDis1，tempp1改為當下外層for迴圈的指標，tempDis1再歸零再進行下一輪，整個跑完最後的tempp1放入rePos[2]。(第3個點要離第1個點和第2個點距離和最近)，最後回傳rePos(三個初始點的索引)

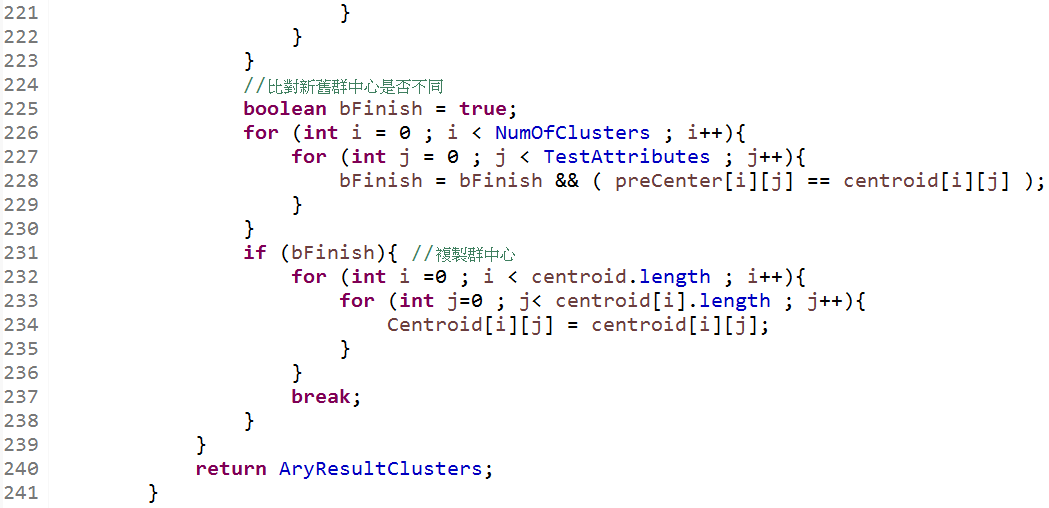
* EuclideanDistance(double[][] centroid)



1. 第164行: EuclideanDistance(double[][] centroid)為一個函式，引數為一個double二維陣列(將資料筆計算歐幾里得距離並且分群)。
2. 第165~188行: 利用巢狀for迴圈，最外層是資料筆數，中間層是分群數，最內層是屬性數，Distance2Centroid接收每一筆資料分別與每個群的質心計算距離，if迴圈當群集的index(中間層for迴圈指標)為0時tmpDis接收Distance2Centroid，idxCluster為0 ; 若else(index不為0)時，內層if迴圈若tmpDis大於Distance2Centroid，則tmpDis接收Distance2Centroid，idxCluster為當下群集的index(中間層for迴圈指標)，最後裡面兩層for迴圈出來，宣告整數型態變數ClusterNumber值為ClusterNo[idxCluster]，AryResultClusters[idxCluster][ClusterNumber]為i+1(當下第幾筆資料，索引值加1)，ClusterNo[idxCluster]加1後存入。

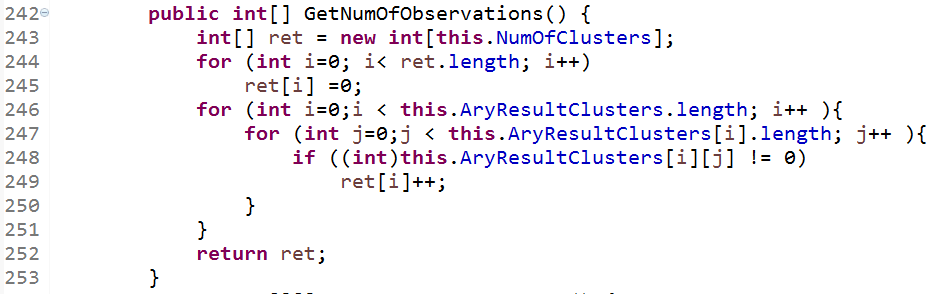
* DoPartition()





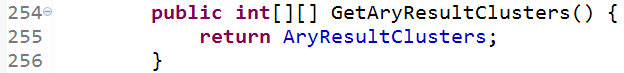
1. 第190~191行: 宣告兩個double型態二維陣列(centroid,Centroid)，兩個陣列大小為分群數，長度為資料筆屬性數。
2. 第192行: 宣告一個整數型態陣列(grandCent)，接收GetStartingPointArrayIndex()所回傳的整數陣列。
3. 第193~201行: 將grandCent取出來的index所對應的座標存到centroid陣列中。
4. 第202行: 宣告一個double型態二維陣列(preCenter)，陣列大小為分群數，長度為資料筆屬性數。
5. 第203行: while迴圈，當true時進行迴圈內動作，false跳出迴圈。
6. 第204行: AryResultClusters陣列設定大小為分群數，長度為資料筆數。
7. 第205~206行: for迴圈將ClusterNo都設為0。
8. 第207行: 將centroid二維陣列作為引數傳入EuclideanDistance()計算。
9. 第208~210行: 利用巢狀for迴圈，外層是分群數，內層是屬性數，由preCenter接收centroid(將目前群中心暫存到preCenter，以做比對用)。
10. 第211~222行: 利用巢狀for迴圈，最外層是分群數，中間層是屬性數，最內層每一群的資料筆數，宣告整數型態變數kNum接收ClusterNo[i]每一群的個數，再將每一群中的所有資料座標每一個屬性分別相加後再分別除以每一群的資料筆數，最後形成新的質心。
11. 第225行: 宣告布林型態變數bFinish值為true。
12. 第226~230行: 利用巢狀for迴圈，外層是分群數，內層是屬性數，比較preCenter和centroid是否一樣，若是bFinish值為true，不是bFinish值為false。
13. 第231~238行: if迴圈，若bFinish值為true，將centroid存到Centroid後break跳出while迴圈。
14. 第240行: 回傳AryResultClusters陣列。

* GetNumOfObservations()



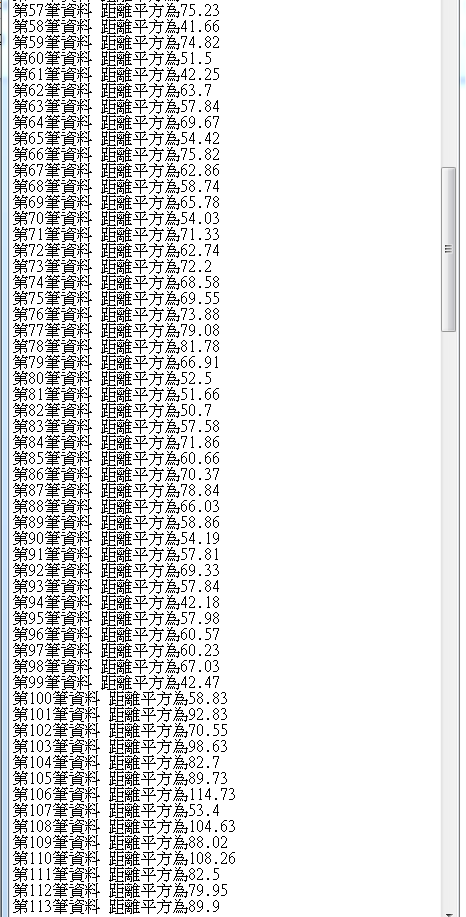
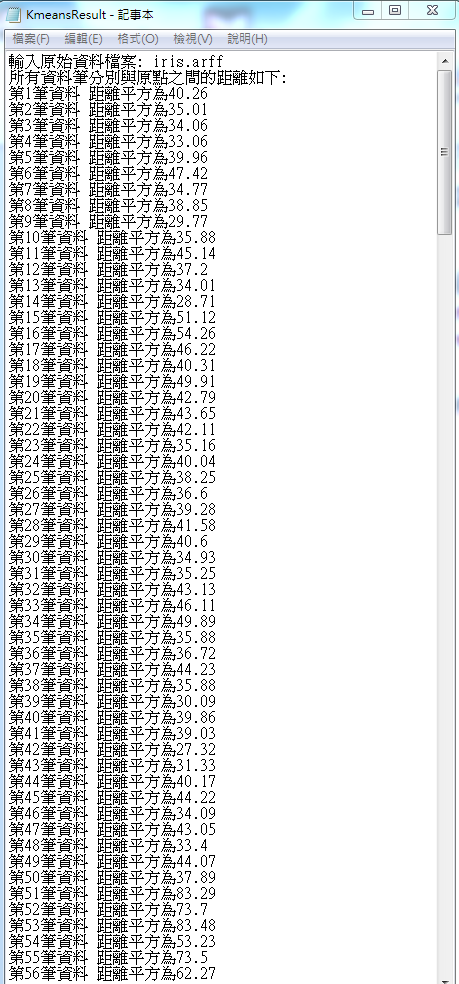
1. 第242行: GetNumOfObservations()為一個函式，回傳型態為整數陣列(每一群的資料筆數)。
2. 第243行: 宣告一個整數型態陣列(ret)，陣列大小為分群數。
3. 第244~245行: for迴圈將ret都設為0。
4. 第246~251行: 利用巢狀for迴圈，外層是AryResultClusters的大小，內層是AryResultClusters的長度，if迴圈判斷若AryResultClusters[i][j]不為0時，ret[i]加1。
5. 第252行: 回傳ret陣列。

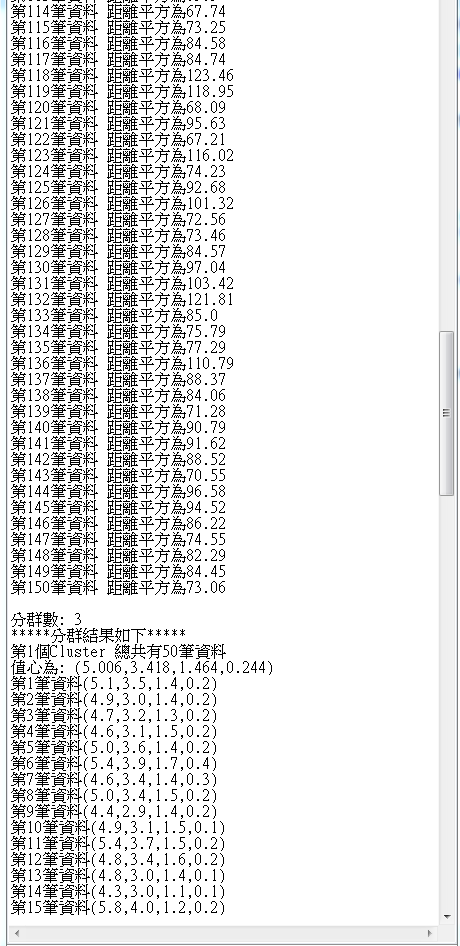
* GetAryResultClusters()



1. 第254~256行: GetAryResultClusters()為一個函式，回傳型態為整數二維陣列(每一群集的資料筆編號)。

* 執行結果





* 原始資料為三群，每群都是50筆資料，總共150筆資料。
* 程式執行結果發現分三群時有一群可以分得很好，另外兩群會有混在一起的狀況。